DOCUMENT RESUME

ED 168 557 IR 007 092

AUTHOR Dageforde, Mary L.
TITLE The BASIC Instructional Program: Conversion into MAINSAIL Language.
INSTITUTION Stanford Univ., Calif. Inst. for Mathematical Studies in Social Science.
SPONS AGENCY Navy Personnel Research and Development Center, San Diego, Calif.
REPORT NO NPRDC-TN-78-11
PUB DATE Apr 78
CONTRACT N-00123-76-C-1543
NOTE 15p.; For related documents, see IR 007 092-096

EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS *Computer Based Laboratories; *Computer Science Education; Higher Education; *Instructional Programs; *Program Development; Programing; *Programing Languages; Tutorial Programs
IDENTIFIERS *MAINSAIL; SAIL

ABSTRACT

This report summarizes the rewriting of the BASIC Instructional Program (BIP) (a "hands-on laboratory" that teaches elementary programming in the BASIC language) from SAIL (a programming language available only on PDP-10 computers) into MAINSAIL (a language designed for portability on a broad class of computers). Four sections contain (1) a description of BIP and reasons for wanting to rewrite it in a more machine-independent language; (2) a discussion of the process of rewriting BIP into MAINSAIL, and in particular, the redesign that was necessary to make BIP more machine-independent; (3) a description of MAINSAIL improvements over the SAIL version; and (4) the current status of MAINSAIL, BIP, and possible future developments. (Author/CMV)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

ED168557

U S DEPARTMENT OF HEALTH,
EDUCATION & WELFARE
NATIONAL INSTITUTE OF
EDUCATION

THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL NATIONAL INSTITUTE OF EDUCATION POSITION OR POLICY

Technical Note 78-11

April 1978

THE BASIC INSTRUCTIONAL PROGRAM: CONVERSION INTO MAINSAIL LANGUAGE

Mary L. Dageforde

Institute for Mathematical Studies in the Social Sciences
Stanford University
Stanford, California 94305

Reviewed by
John D. Ford, Jr.

Navy Personnel Research and Development Center
San Diego, California 92152

007092

FOREWORD

This research and development was conducted in response to the Navy Decision Coordinating Paper, Education and Training Development (NDCP-Z0108 PN) under subproject Z0108-PN.32, Advanced Computer-Based Systems for Instructional Dialogue, and the sponsorship of the Director, Naval Education and Training (OP-99). The overall objective of the subproject is to develop and evaluate advanced techniques of individualized instruction.

This is one in a series of six reports dealing with the BASIC (Beginner's All-Purpose Symbolic Instruction Code) Instructional Program (BIP), which is a "tutorial" programming laboratory designed for the student who has had no previous training in programming. The others concern the BIP student manual (Note 1, 1978), supervisor manual (Note 2, 1978), system documentation (Note 3, 1978), conversion of the student manual into the MAINSAIL language (Note 4, 1978), and curriculum information networks for computer-assisted instruction (Beard, Barr, Gould, & Wescourt, 1978). This report describes the conversion of BIP itself into the MAINSAIL programming language.

The work was performed under Contract N000123-76-C-1543 to Stanford University. The contract monitors were Dr. John D. Fletcher and Dr. James D. Hollan.

J. J. CLARKIN
Commanding Officer

SUMMARY

The BASIC Instructional Program (BIP) is a "hands-on laboratory" that teaches elementary programming in the BASIC language. This report summarizes the rewriting of BIP from SAIL, a programming language available only on PDP-10 computers, into MAINSAIL, a language designed for portability on a broad class of computers.

The introductory section describes BIP and tells the reason for wanting to rewrite it in a more machine-independent language. Section 2, the main body of the report, discusses the actual rewriting of BIP in MAINSAIL, and in particular, the redesign that was necessary to make BIP more machine-independent. Section 3 describes MAINSAIL improvements over the SAIL version, and Section 4, the current status of MAINSAIL, BIP, and possible future developments.

CONTENTS

	Page
SECTION 1. INTRODUCTION	1
1.1 Description of the BASIC Instructional Program	1
1.2 Original Plan: To Rewrite in BASIC	2
1.3 Final Language Decision: MAINSAIL	3
SECTION 2. THE REWRITING OF BIP INTO MAINSAIL	5
2.1 Initial Code Compatible with Both SAIL and MAINSAIL Compilers	5
2.2 The Main Data Structures	5
2.3 File Names	6
2.4 Space Saving	7
2.5 History Files	7
2.6 Data and Time Information	7
SECTION 3. IMPROVEMENTS OVER THE SAIL VERSION OF BIP	9
SECTION 4. CURRENT STATUS AND POSSIBLE FUTURE DEVELOPMENTS	11
REFERENCES	13
REFERENCE NOTES	13

SECTION 1. INTRODUCTION

The BASIC Instructional Program (BIP) is a sophisticated computer-based laboratory for instruction in elementary programming in the BASIC language. It was developed as an experimental system on the IMSSS PDP-10 research computer facility in a specialized high-level programming language called SAIL (Reiser, 1976), which is presently available only to the PDP-10 user community. Thus, a proposal was made to rewrite BIP in a more machine-independent language, so that it could be implemented on other (notably, smaller) systems.

1.1 Description of the BASIC Instructional Program

BIP is a "tutorial" programming laboratory designed for students who have had no previous training in programming. These students work their problems at an interactive terminal under the constant supervision of the instructional program, which offers hints or assistance at their request. BIP monitors each student's programming attempts, continually updates its model of the student's knowledge, and uses an information network representation of the curriculum to allow individualized selection of the programming problems given to the student. Since BIP runs in an interactive environment, the student receives immediate feedback about any syntax errors and information about other errors as soon as they are detected. These features keep the student from going too far down the wrong track without some warning. BIP's interpreter is built into the instructional program so that further assistance is available after the first error message.

BIP has been used successfully by students at Stanford, at the U.S. Naval Academy, and at local colleges with access to the IMSSS instructional facility.

The BIP system, as developed in SAIL and described in Barr, Beard, and Atkinson (1976), is composed of ten major subsystems:

1. BASIC Interpreter--Responsible for identifying syntax and runtime errors. Includes standard debugging and editing commands.

2. Program error analyzer (ERROR DOCTOR)--Identifies the structural bugs in student programs.

3. HELP system--Gives advice and references to the off-line student manual based on cues from the first two systems.

4. Solution Checker--Verifies that the student's program does in fact solve the problem.

5. The curriculum and the BIP Student Manual--A large set of programming problems, with hints and model solutions, and the off-line manual that students use as a reference during the course.

6. Curriculum Information Network and student model--Describes the structure of the curriculum and current abilities of the student, so that task sequencing can be individualized.

7. Instructor/Supervisor capabilities--Examples are enrolling students, monitoring their progress, sending messages, and responding to complaints.

8. The graphics debugging package (FLOW) and graphics flowcharting option (REP).

9. The data collection facility--Used to save "protocols," or records of student sessions.

10. The student file storage system--Used to allow students to save up to ten of their programs at a time.

FLOW and REP (8 above) are the only parts of SAIL BIP that have not yet been written for MAINSAIL BIP. Before they can be implemented, a machine-independent display package must be written to access terminal-dependent procedures.

1.2 Original Plan: To Rewrite in BASIC

The original proposal was to rewrite the BIP instructional system in the BASIC programming language, the same language that BIP teaches. Since BASIC is available on more computer systems than any other programming language, this would allow the test and evaluation of the BIP course in a wide community of users. Further, BASIC is easy to implement on a small (relatively cheap) computer, fully interactive, and fairly easy to learn.

The BASIC language, however, is not well suited for the development of programs as large and complex as the BIP system. It has neither the versatility nor the power of ALGOL-like languages (e.g., SAIL and LISP). To implement the instructional laboratory in BASIC would require considerable modification and redesign.

Another difficulty in trying to rewrite BIP in BASIC is that, although it is available on almost all general-purpose computers, most manufacturers have developed a unique dialect of the language. This has caused serious problems in transporting even small BASIC programs from one site to another, since syntax changes are almost always necessary. Changes of this nature to a program as large as the BIP system would make a truly transportable system unfeasible.

Finally, there are considerable difficulties involved in rewriting the code in BASIC for a small computer, primarily because of the restrictions of the BASIC language (no recursion, no string scanning, and small segment size). BIP's Curriculum Information Network and the student model (which form the data base of the task-selection algorithm) make extensive use of the associative data structures available in the SAIL language; and the syntax parser is thoroughly recursive. These subsystems would be very difficult to implement in BASIC.

1.3 Final Language Decision: MAINSAIL

MAINSAIL (MAchine-INdependent SAIL) (Wilcox, 1977a) is a language being developed at the Stanford University Medical Experimental (SUMEX) Computer Facility. MAINSAIL has evolved from SAIL, which was developed in the late 1960's at Stanford's Artificial Intelligence Laboratory. SAIL was designed for execution on a PDP-10 computer with the TOPS-10 operating system. MAINSAIL, as reflected in its name, provides capabilities similar to those in SAIL independent of the underlying computer system (Wilcox, 1977b). It is designed to be powerful and efficient, with a high degree of portability on a broad class of computers. In fact, if design goals are met, any program written in pure MAINSAIL will be able to be run on any computer (including some minicomputers) that has a MAINSAIL compiler--regardless of differing word sizes, operating systems, numbers and types of registers, etc. It is designed for the development of portable software, such that programs can be transported, with no alteration, among all implementations.

Any general-purpose computer with a reasonably powerful instruction set, suitable data formats and addressability, sufficient address space, a file system, and an interactive terminal, should be capable of supporting an implementation of MAINSAIL. MAINSAIL is currently running on PDP-10's with both TENEX and TOPS-10 operating systems, and is under development for PDP-11's with three operating systems: RT11, RSX11, and UNIX. A number of other implementations (e.g., TI-990, HP-3000, IBM-370, INTERDATA) are under consideration.

Because it is similar to SAIL, and because it was designed expressly for portability, MAINSAIL was found to be ideally suited for the translation of BIP. Any computer that supports MAINSAIL will be able to run the MAINSAIL version of BIP without modifications.

SECTION 2. THE REWRITING OF BIP INTO MAINSAIL

2.1 Initial Code Compatible with Both SAIL and MAINSAIL Compilers

When the MAINSAIL translation was begun in October 1976, MAINSAIL's first implementation, on a PDP-10 (the computer on which the translation was done), was not yet completed. Therefore, for several months, it was necessary to write code that was acceptable to both the SAIL and the MAINSAIL compilers. This was accomplished by writing pure MAINSAIL code as much as possible, but using macros and procedures to describe to the SAIL compiler the SAIL equivalents of various MAINSAIL system procedures, reserved words, etc. Conditional compilation was also used occasionally; that is, sometimes two segments of code would be written to do the same type of thing, one for the SAIL compiler and the other for the MAINSAIL compiler (a macro that was set just before using either compiler determined which segment of code was compiled).

Thus, the SAIL compiler "understood" the MAINSAIL code, and the modules that were written during the first few months of the translation (notably, the parser and the interpreter) were successfully compiled and tested by the SAIL compiler. Then, in April 1977, when MAINSAIL was up on the PDP-10, BIP was compiled and run more often with the MAINSAIL compiler. It was still occasionally compiled with SAIL because of the availability of a good symbolic debugger called SAIL (Reiser, 1976). (A debugger for MAINSAIL is now being written.) However, since May 1976, the MAINSAIL compiler has been used exclusively, and most of the code written since then has been solely for the MAINSAIL compiler.

2.2 The Main Data Structures

BIP was written before the record data structure capability was added to SAIL, at a time when *LEAP* was the only facility available for data storage and retrieval other than arrays. As mentioned in the Introduction, SAIL BIP made extensive use of *LEAP* structures that were replaced by records in the MAINSAIL version.

LEAP provides high-level data structures, such as sets and lists, and operations on those structures, such as union, intersection, removal and insertion of elements, etc. *LEAP* also has an associative data store of triples that allows one to store and retrieve information based on relations between specified elements. Take, for example, a "Skill of" relation. If the programmer has formed associations such as "Skill of Task9 is 2" and "Skill of Task9 is 8," etc., he can then perform searches to find the set of skills in task 9 by having the *LEAP* search routines return a set of triples whose first component is "Skill of" and whose second component is "Task9."

Records provide a means by which a number of closely related variables may be allocated and manipulated as a unit, without the overhead or limitations associated with using parallel arrays and without the restriction that the variables all be of the same data type. Each record is an instance of a user-defined record class, which serves as a template describing the various fields of a record. In the example above, instead of forming a "Skill of Task9" association for every skill in task 9, and then later asking *LEAP* to form a set of the skills in task 9 by searching for all triples whose first

component is "skill of" and whose second component is "task#," we could build up and keep the "set" readily available somewhere, by putting the task 0 skills in a linked list of records. A record class called "skill" could be defined to have two fields, one for a skill number and the other for a pointer to the next record in a linked list of skills. We could form a record of the class "skill" for each skill in task 0, and if the skills are 2, 5, and 12, then the linked list could be pictured as

One of the main data structures in MACONACO.SOP is a linked list of records, one for each task. The thirteen fields of each record hold various information needed by SOP, including the task name, its number, the location in the SKOOL file of its description, model solution, and hints, a pointer to the linked list of its skills (as described above), etc. See Section 2.2 of Vote 3 (1978) for a more detailed description of the task records.

LISP was used extensively throughout the SAIL version of SOP, not necessarily because it was the best of all programming structures for what one needed, but because it was the only major data storage and retrieval method, other than arrays, available in SAIL at the time. The high-level LISP structures have to be very general--usually more general (and thus less efficient) than needed in any particular instance in SOP, and the LISP syntax is often confusing and obscure. MACONACO does not support LISP because the runtime code for it would be excessive and because most of the capabilities provided by LISP are easily implemented with records. The use of records instead of LISP in the MACONACO version of SOP did not cause any loss in clarity, simplicity, or flexibility--in fact, it led to a number of simplifications, both conceptually and in implementation.

2.3 File Names

File names on different computers have different length and starting character specifications. On some computers, they may be a maximum of six characters long; on others they may be almost as long as the user desires. Some computers allow the starting character to be either a number or a letter, whereas others specify that it must be only a letter. Thus, in order for the MACONACO SOP file names to be truly "machine-independent," they all begin with letters and are a maximum of six characters in length.

The format of student history file names, for example, is <student number>.HST in the SAIL version of SOP. In the MACONACO version, it is HST<student number>. (Note that 999 is the highest possible student number, since a four-digit number would make the history file name seven characters long.)

As another example, students are allowed to save up to ten of their programs for later use. The file names for these in SAIL SOP is

<student number>.<whatever name the student assigns to the file>.

In the [illegible] version, students may still assign whatever names they wish to their programs, but the file names under which the programs are actually saved are completely different. They are of the format [illegible] where [illegible] is a number between 0 and 9. A new element may called profile while the names assigned by the students. For example, if student number 00 saves his first program under the name 'PROGRAM1', then profile [1] = 'PROGRAM1' and the name of the file actually saved is [illegible]. Between sessions, the student-assigned names are saved in the student history (Section 5.3 of Note 5, 1978).

3.8 Space Saving

The [illegible] version of BIP requires a very large space, in particular for storage. All the model solutions and task descriptions for the BI programming tasks are kept in memory-resident string arrays. In are all the error messages (nearly 100), the hint messages (nearly 100, some of which are 500-600 characters long), and one-line descriptions of each of BI programming skills. The BIP-11 has enough space for these arrays, but many other computers, especially microcomputers, do not. Therefore, space-saving measures were taken in the [illegible] version of BIP.

Instead of being stored in string arrays, all the help messages, error messages, and skill descriptions are on a file called BSKD. Each group of messages is on a separate page of BSKD, in order. The beginning of each page has pointers to the start of all the messages on that page. Given a page and a message number, a specific BIP procedure will copy the appropriate message from the file BSKD.

All of the task information--task descriptions, hints, model solutions, etc.--is kept on the file TASKS, and the curriculum data structure (Section 5.2 of Note 5, 1978) contains pointers into that file so that appropriate information can be efficiently accessed when needed.

3.9 Student Files

Each individual student's personal history file is a data file used to store information about the student's current state (what task he is currently working on, how many tasks completed so far, etc.), and past performance on tasks and skills. [illegible] BIP uses binary words to save information in the history for each task and for each skill. The binary encryption is clearly machine-dependent, since some computers have 16-bit words, some 32-bit words, etc. [illegible] has a data type called bits for representing a short sequence of bits. It will support at least 16 bits (independent of the word size of the host machine), and that is the number used in word elements of the student history in [illegible] BIP.

3.10 Date and Time Information

Saving date and time information, such as the amount of time spent on each task, the date of the last edit on, etc., would be useful and of interest to the BIP supervisor. This was done in the [illegible] version of BIP. Again, this type of information is accessed in a machine-dependent manner, using [illegible] calls. For the [illegible] version, two simple machine-dependent

[illegible]

SECTION 3. IMPROVEMENTS OVER THE SAIL VERSION OF BIP

The fact that the MAINSAIL version of BIP is machine-independent, while the SAIL version is limited to a PDP-10 with a TENEX operating system, and yet does all that the SAIL version does (excluding the graphic programming aids) is of course itself a tremendous improvement. But a number of other improvements in efficiency and design were also made. Analysis of the SAIL version prior to writing the MAINSAIL version sparked ideas for improved code and algorithms and revealed some bugs, which were promptly fixed.

The SAIL version of BIP was written, expanded, and improved by a number of people over a period of years. The project described in this report provided an ideal opportunity for one person to analyze the system developed by those people, learning about its design, its strengths and weaknesses, and then to rewrite it all, using what had been written as a base and yet redesigning algorithms and data structures where improvements could be made (or where changes had to be made for machine-independence). Important space-saving measures, as described in Section 2.4, were taken, and virtually every procedure had minor improvements in code that contributed to the overall improvement in efficiency and design.

Major redesign was required in order to use record data structures in place of the extensively-used associative data structures (LEAP) in the SAIL version. The new design could also be considered an improvement in clarity and ease of implementation. An extreme example is the ERRDOK module. The SAIL version used a number of complicated LEAP structures, most of which were not related to the actual function of the ERRDOK procedures. (At the time that ERRDOK was written, there was some hope of using its complicated information about the structure of the student's program in a more sophisticated solution checker.) Only two small linked lists of records are used in the MAINSAIL version to perform the same functions actually implemented in the SAIL version (see Section 6 of Note 3, 1978).

SECTION 4. CURRENT STATUS AND POSSIBLE FUTURE DEVELOPMENTS

All the subsystems in the SAIL version of BIP that were listed in Section 1.1, except for the terminal-dependent graphics features, have been implemented in MAINSAIL. MAINSAIL BIP is capable of running on all PDP-10 computers that have either TENEX or TOPS-10 operating systems, and it will be put up on PDP-11's when MAINSAIL is available on them. It should be able to run on any computer that has MAINSAIL.

The BIP Student Manual (Note 4, 1978) has been modified to conform with the MAINSAIL version of BIP. Although BIP is a "self-instructional" system, a Supervisor's Manual has been written to explain the goals, methods, and operation of the system to supervisory personnel (Note 2, 1978). Also, it includes documentation of all necessary clerical capabilities, such as registering and terminating students, examining their progress, and adding special programming tasks.

Complete documentation has been written to describe all the programming modules that comprise the MAINSAIL BIP system including the BASIC interpreter, the task-selection algorithm, the curriculum, the solution checker, the help system, and collection of relevant student performance data (Note 3, 1978). A detailed description of the information saved in student histories and of the curriculum data structures is also included.

Future possible additions and improvements would include making MAINSAIL BIP available on various other machines, and writing a graphics display package and terminal-dependent procedures so that the graphics debugging package (FLOW) and the graphics flow-charting option (REP) could be implemented.

REFERENCES

Barr, A., Beard, M., & Atkinson, R. C. The computer as a tutorial laboratory: The Stanford BIP project. International Journal of Man-Machine Studies, 1976, 8, 567-596.

Beard, M., Barr, A. V., Gould, L., & Wescourt, K. Curriculum information networks for computer-assisted instruction (NPRDC TR 78-18). San Diego: Navy Personnel Research and Development Center, April 1978.

Reiser, J. SAIL users manual (Artificial Intelligence Memo 289). Stanford, CA: Stanford Artificial Intelligence Laboratory, Stanford University, 1976.

Wilcox, C. R. MAINSAIL language reference manual. Stanford, CA: SUMEX Computer Project, Stanford University Medical Center, 1977 (a).

Wilcox, C. R. The MAINSAIL project: Developing tools for software portability. Proceedings of the First Annual Symposium on Computer Application in Medical Care, IEEE Catalog No. 77 CH1270-8C, pp. 76-83, Washington, DC, 1977 (b).

REFERENCE NOTES

1. Beard, M. H., & Barr, A. V. The BASIC instructional program student manual (NPRDC Special Rep. 77-2). San Diego: Navy Personnel Research and Development Center, October 1976.

2. Dageforde, M. L., & Beard, M. H. The BASIC instructional program: Supervisor's manual (NPRDC Tech. Note 78-10). San Diego: Navy Personnel Research and Development Center, April 1978.

3. Dageforde, M. L. The BASIC instructional program: System documentation (NPRDC Tech. Note 78-12). San Diego: Navy Personnel Research and Development Center, April 1978.

4. Dageforde, M. L., Beard, M. H., & Barr, A. V. The BASIC instructional program student manual: MAINSAIL conversion (NPRDC Tech. Note 78-9). San Diego: Navy Personnel Research and Development Center, April 1978.